Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 1

Тема Построение и программная реализация алгоритма полиномиальной интерполяции табличных функций.

Студент Никуленко И.В.

Группа ИУ7-42Б

Оценка (баллы) _____

Преподаватель Градов В.М.

Москва.
2021 г

Цель работы. Получение навыков построения алгоритма интерполяции таблично заданных функций полиномами Ньютона и Эрмита.

1 Исходные данные

1. Таблица функции и её производных

x	y	y'
0.00	1.000000	--1.000000
0.15	0.838771	-1.14944
0.30	0.655336	-1.29552
0.45	0.450447	-1.43497
0.60	0.225336	-1.56464
0.75	-0.018310	-1.68164
0.90	-0.278390	-1.78333
1.05	-0.552430	-1.86742

2. Степень аппроксимирующего полинома – n.

3. Значение аргумента, для которого выполняется интерполяция.

2 Код программы

Код программы представлен на листингах 1-2.

Листинг1. cogs.py

```
from math import fabs, ceil


def table_output(table):
    print("+{:7s}|{:11s}|{:11s}+".format('-', '-', '-').replace(' ', '-'))
    for i in table:
        print('|{:^7.2f}|{:^11.6f}|{:^11.6f}|'.format(i[0], i[1], i[2]))
        print("|{:7s}|{:11s}|{:11s}|".format('-', '-', '-').replace(' ', '-'))
```

```
def point_selection(table, n, x):
    table_size = len(table)
    closest_point_indx = min(range(table_size), key=lambda i: fabs(table[i][0] - x))
    req_space = ceil(n/2)
    if closest_point_indx + req_space + 1 > table_size:
        start = table_size - n
        end = table_size
    elif closest_point_indx < req_space:
        start = 0
        end = n
    else:
        start = closest_point_indx - req_space
        end = start + n
    result = table[start:end]
    return result


def newton(orig_table, n, x):
    table = []
    for i in orig_table:
        table.append(i[:])
    table = point_selection(table, n, x)
    for i in range(1, n):
        for j in range(n - 1, i - 1, -1):
            table[j][1] = (table[j][1] - table[j - 1][1]) / (table[j][0] - table[j - i][0])
    result = 0
    for i in range(n):
        temp = table[i][1]
        for j in range(i):
            temp *= (x - table[j][0])
        result += temp
    return result


def hermite(orig_table, n, x):
    table = []
    for i in orig_table:
        table.append(i[:])
        table.append(i[:])
    table = point_selection(table, n, x)
    for i in range(n - 1, 0, -1):
        if fabs(table[i][0] - table[i - 1][0]) < 1e-7:
            table[i][1] = table[i][2]
        else:
            table[i][1] = (table[i][1] - table[i - 1][1]) / (table[i][0] - table[i - 1][0])
    for i in range(2, n):
```

```
            for j in range(n - 1, i - 1, -1):
                table[j][1] = (table[j][1] - table[j - 1][1]) / (table[j][0] - table[j - i][0])
    result = 0
    for i in range(n):
        temp = table[i][1]
        for j in range(i):
            temp *= (x - table[j][0])
        result += temp
    return result


def xy_swap(table):
    new_tab = []
    for i in table:
        new_tab.append(i[:])
    for i in new_tab:
        i[0], i[1] = i[1], i[0]
    return new_tab
```

Листинг2. main.py

```
from cogs import *


def main():
    tab = [[0.00, 1.0, -1.0],
           [0.15, 0.838771, -1.14944],
           [0.30, 0.655336, -1.29552],
           [0.45, 0.450447, -1.43497],
           [0.60, 0.225336, -1.56464],
           [0.75, -0.018310, -1.68164],
           [0.90, -0.278390, -1.78333],
           [1.05, -0.552430, -1.86742]]
    x = 0.525

    table_output(tab)
    print('\n\nx = 0.525:')
    print("+{:9s}|{:11s}|{:11s}|{:11s}|{:11s}+".format('-', '-', '-', '-', '-').replace(' ', '-'))
    print('|         |   n = 1   |   n = 2   |   n = 3   |   n = 4   |')
    print("|{:9s}|{:11s}|{:11s}|{:11s}|{:11s}|".format('-', '-', '-', '-', '-').replace(' ', '-'))


    print('| Ньютон  |', end='')
    for i in range(1, 5):
        print('{:10.6f} |'.format(newton(tab, i + 1, x)), end='')
    print("\n|{:9s}|{:11s}|{:11s}|{:11s}|{:11s}|".format('-', '-', '-', '-', '-').replace(' ', '-'))
```

```
        print('| Эрмит   |', end='')
        for i in range(1, 5):
            print('{:10.6f} |'.format(hermite(tab, i + 1, x)), end='')
        print("\n|{:9s}|{:11s}|{:11s}|{:11s}|{:11s}|".format('-', '-', '-', '-', '-').replace(' ', '-'))

        root_tab = sorted(xy_swap(tab), key=lambda i: i[0], reverse=True)
        print('Корень функции:')
        print("+{:11s}|{:11s}|{:11s}|{:11s}+".format('-', '-', '-', '-', '-').replace(' ', '-'))
        print('|   n = 1   |   n = 2   |   n = 3   |   n = 4   |')
        print("|{:11s}|{:11s}|{:11s}|{:11s}|\n|".format('-', '-', '-', '-', '-').replace(' ', '-'), end='')
        for i in range(1, 5):
            print('{:10.6f} |'.format(newton(root_tab, i + 1, 0)), end='')
        print("\n|{:11s}|{:11s}|{:11s}|{:11s}|".format('-', '-', '-', '-', '-').replace(' ', '-'))


if __name__ == '__main__':
    main()
```

3 Результаты работы

1. Значения y(x) при степенях полиномов Ньютона и Эрмита n = 1, 2, 3 и 4 при фиксированном x = 0.525

x = 0.525:

	n = 1	n = 2	n = 3	n = 4
Ньютон	0.337891	0.340419	0.340314	0.340324
Эрмит	0.337891	0.340358	0.340323	0.340324

2. Корень заданной таблично функции, найденный с помощью обратной интерполяции, используя полином Ньютона при различных степенях полинома.

Корень функции:

n = 1	n = 2	n = 3	n = 4
0.738727	0.739174	0.739095	0.739088

4 Вопросы при защите лабораторной работы

1. Будет ли работать программа при степени полинома n=0?

Да, в результате выполнения программы будет получено значение узла, ближайшего к аргументу x, для которого выполняется интерполяция.

2. Как практически оценить погрешность интерполяции? Почему сложно применить для этих целей теоретическую оценку?

При достаточно быстром убывании членов ряда, для практической оценки погрешности, можно оставить члены начиная с определенного, его значение будет являться погрешностью.

Для применения теоретической оценки требуются производные интерполируемой функции, которые могут быть неизвестны.

3. Если в двух точках заданы значения функции и ее первых производных, то полином какой минимальной степени может быть построен на этих точках?

Имеется 4 узла, из этого следует, что минимальная степень полинома равна 3.

4. В каком месте алгоритма построения полинома существенна информация об упорядоченности аргумента функции (возрастает, убывает)?

При выборе узлов, наиболее близких к аргументу. Упорядоченность аргументов позволяет затрачивать меньше ресурсов на поиск узлов.

5. Что такое выравнивающие переменные и как их применить для повышения точности интерполяции?

Для повышения точности интерполяции функцию на протяжении нескольких шагов таблицы приближают к некой элементарной функции. На переменных, полученных из элементарной функции, строится таблица, их интерполяция и нахождение обратным преобразованием. Эти переменные называются выравнивающими.